# Exhibit 18

# AutobahnFX Java API

Version: prod-8-3-5

Issued: Dec-2008

## Version 8-3 release notes:

- The API is now able to fail over automatically to another connection URL if the connection to the server is lost. Please find details of the reconnection logic below.
  - FxApi#connect() method uses the connection URL as specified in 'realm.url' property.
  - Once a connection is established, the API gets a list of other connection URLs available for the same connection point.
  - If the connection to the server is lost, the API reports UNAVAILABLE status and tries to re-connect to the other URLs. The client code does not need to perform any actions, the reconnection is done automatically by the API. CONNECTED status is reported after a working connection URL is found. All price subscriptions are restored automatically.
  - DISCONNECTED status is reported only if the session is forced to be shut down from DB side which normally happens on weekends. The client code should issue connect() method to re-estabilsh the session.
  - If the URL provided to the API in 'realm.url' property is down than FxApi#connect() method will fail. If you upgrade the API installation to 8-3 version then FxApi#connect() method will use all connection URLs updated from the server in the previous session.
- Connection URLs are updated on API download site. Please check that all connection URLs available for your connection point are configured on the firewall.
  - https://dbfxb2b.fxmarkets.com/fxratesapi/ (login: fxratesapi password: ratesapiversions)

## Version 8-2 release notes:

- In 8-2 release Deutsche Bank will start pricing selected currency pairs with greater precision of tenth of a pip.
- It is mandatory that all clients update their API installations to 8-1 version prior to the release. All versions below 8-1 will stop working at the release date.
- New asynchronous trading methods added in 8-1 version are now supported.
- New method `subscribeCustomSpotRoll` is added to FxApi interface. The method allows to book a swap deal that rolls a specified spot position to a forward date. Please refer to `FxApi` javadoc for more details.
- Client code is no longer required to call `TradeResult#acknoledge()` on a trade result received. The method is marked as deprecated.

## Version 8-1 release notes:

- The main feature of 8-1 API release is an ability of API to update itself automatically once a new API version is available
- Later this year, with 8-2 release in July, Deutsche Bank will conduct a major upgrade of the Autobahn FX platform providing more precise and robust pricing to the customers. It is mandatory that all clients update their API installations to 8-1 version prior to 8-2 release. After July release the 8-1 API installations will update themselves to 8-2 version, all versions prior to 8-1 will stop working at this point. Please contact the Client Integrations team for more details.
- New trading methods are added to FxApi interface and documentation. The methods will be supported in 8-2 version and marked with @since 8-2 tag in javadoc. The new methods submit trade request asynchronously without blocking the thread. The trade result is reported to the client application via a `TradeResultListener` callback. See FxApi javadoc for more details.
- The new property `FxApi.LOG_LEVEL_DEBUG` is introduced to enable logging of every quote requested from the API in a separate file `abfxquotesdetails.log`. By default, this file will not be created.
- The new property `FxApi.LOG_SIZE` allows to control the total maximum size of all log files generated by the API. This property should have an integer value of megabytes which will be allocated for log files. Please refer to `FxApi` javadoc for details.

| | |
|---|---|
| String | `getTradeId()`<br>If `getStatus()` is `FxApi.TRADE_ACCEPTED` the ID of the trade as allocated by the server is returned. |
| boolean | `isOutright()`<br>Determines whether this is an outright trade. |
| boolean | `isSpot()`<br>Determines whether this is a spot trade. |
| boolean | `isSwap()`<br>Determines whether this is a swap trade. |

# Fields

## TRADING_DISABLED

`public static final java.lang.String` **`TRADING_DISABLED`**

The trade is rejected because the currency pair was marked not tradeable whilst the trade request was 'in flight'.
Constant value: **`TRADING DISABLED`**

**See Also:**
getErrorCode()

## INSUFFICIENT_LIQUIDITY

`public static final java.lang.String` **`INSUFFICIENT_LIQUIDITY`**

The trade is rejected because you do not have sufficient spot and/or forward liquidity to complete the trade. Result of `Quotes.getMaxTradable(buySell, tenor, tradingCurrency)` provides a snapshot of current available liquidity. However the precise amount of available liquidity will also fluctuate with market movements or account configuration changes, and this may happen whilst the trade request was 'in flight'.
Constant value: **`INSUFFICIENT LIQUIDITY`**

**See Also:**
getErrorCode()

## STALE_QUOTE

`public static final java.lang.String` **`STALE_QUOTE`**

The trade is rejected because you attempted to trade on a quote object that has expired or been superseded by a more recent quote. this is most likely indicative of a coding or performance issue in your adaptor. All quotes objects should be processed as quickly as possible, and all trade requests need to be placed against the latest quotes object received. Please contact gm-ecommerce.implementations@db.com for further assistance.
Constant value: **`STALE QUOTE`**

**See Also:**
getErrorCode()

## INVALID_RATE

`public static final java.lang.String` **`INVALID_RATE`**

The trade is rejected because of rate specified is invalid.
Constant value: **`INVALID RATE`**